15.1.2024

Fabian Patzwall

WS 23/24 3. SEMESTER

Version 1.0

Entwicklerdokumentation

Projekt „Smarter Würfel“

Inhalt

[Arbeitsfähig werden 2](#_Toc156291182)

[Hardware vorbereiten 2](#_Toc156291183)

[IDE einrichten 2](#_Toc156291184)

[Die Code Base 5](#_Toc156291185)

[main.py 5](#_Toc156291186)

[lowpower.py – Energiesparen 5](#_Toc156291187)

[dice\_library.py 5](#_Toc156291188)

[Vorbedingungen 6](#_Toc156291189)

[Funktionen 7](#_Toc156291190)

[Funktionsprüfung und Validierung 7](#_Toc156291191)

[Fehlerbehebung 8](#_Toc156291192)

[Zukünftige Entwicklung 9](#_Toc156291193)

[Hardware 9](#_Toc156291194)

[Software 10](#_Toc156291195)

[Ressourcen zur Entwicklung 10](#_Toc156291196)

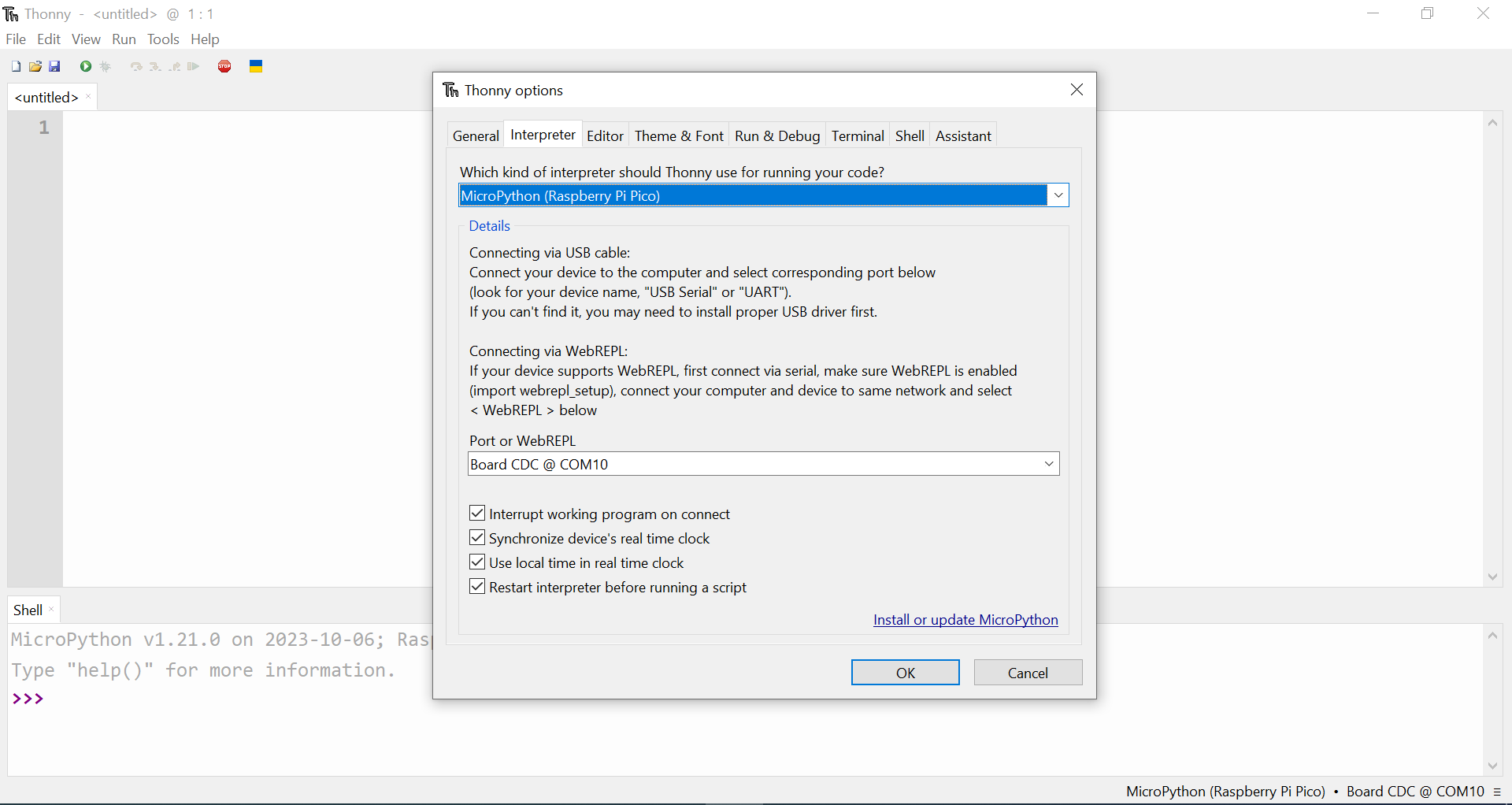
[Kontakt 10](#_Toc156291197)

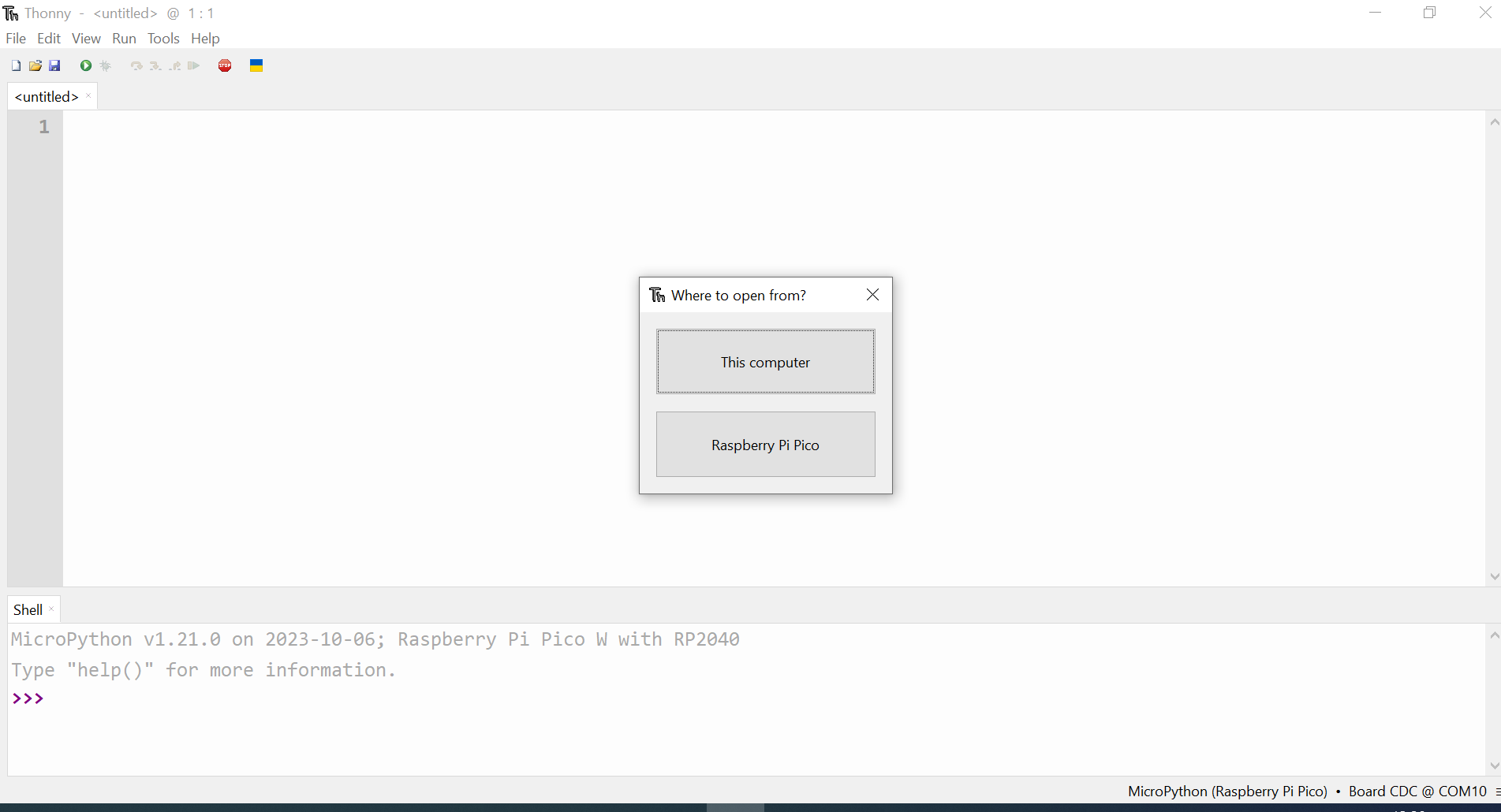
# Arbeitsfähig werden

## Hardware vorbereiten

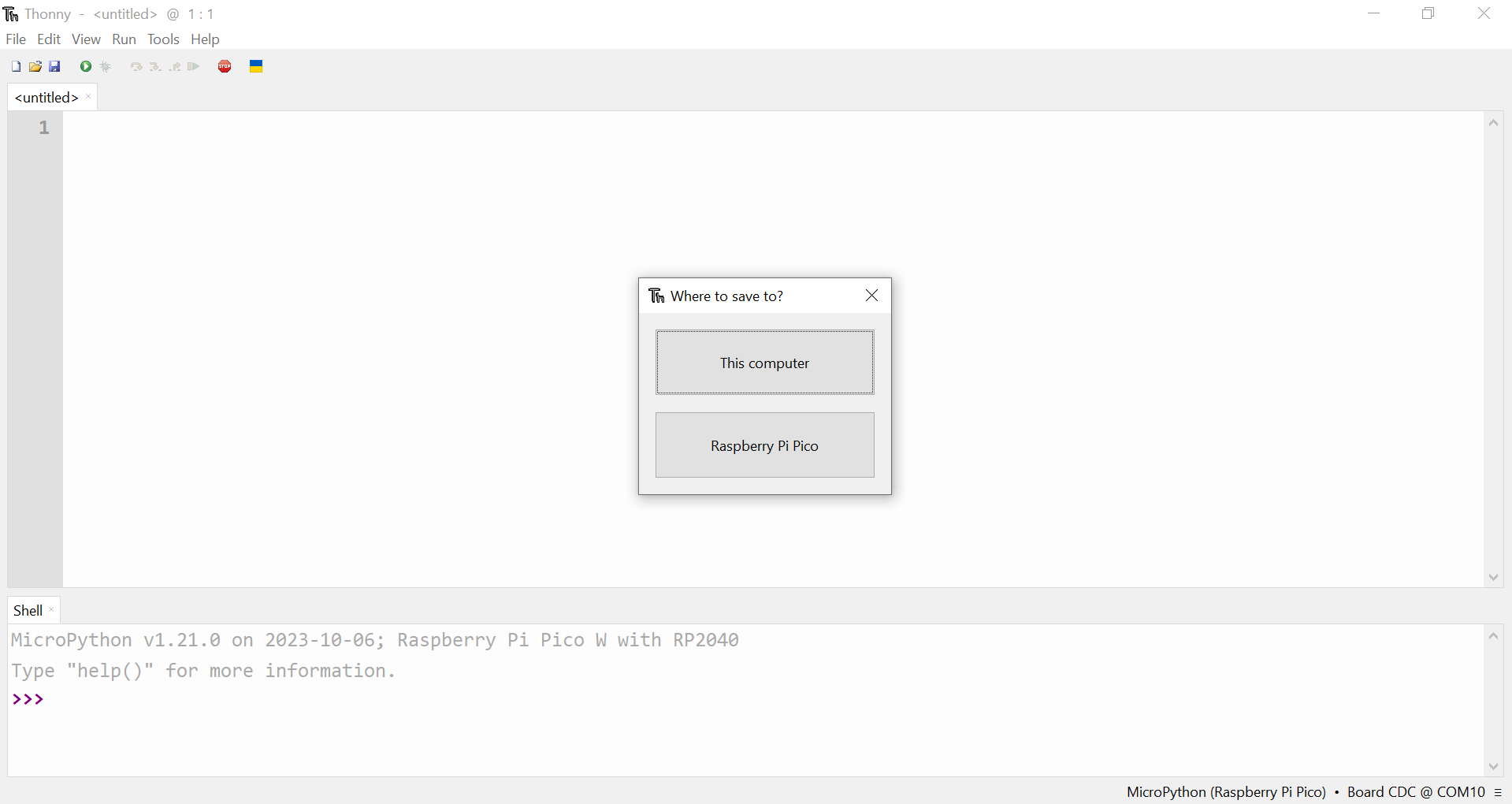
1. Lade die aktuellste microPython Version (.uf2 Datei) für den Raspberry Pi Pico W auf folgender Website runter: <https://micropython.org/download/RPI_PICO_W/>
2. Währed du den Pico an den USB Anschluss eures Rechners anschließt haltet den Bootselect (BOOTSEL) Knopf gedrückt.
3. Zieht die .uf2 Datei auf das vom Pico bereitgestellte Laufwerk auf - damit ist die Installation von MikroPython abgeschlossen.
4. Trenne den Pico von deinem Rechner und verbinde Ihn erneut, ohne den Boot Knopf (BOOTSEL) zu drücken.

## IDE einrichten

1. Lade dir die Thonny DIE runter und installiere die für dein System passende Version. Direktlink: <https://thonny.org/>
2. Öffne das Programm. Wenn du den Raspberry Pi Pico an deinem Laptop angeschlossen hast solltest du folgendes sehen nach dem Öffnen: 
3. Klicke oben links auf das Feld „Tools“. Wähle dort „Options“ aus. Jetzt wähle im Reiter Interpreter MicroPython (Raspberry Pi Pico) aus und bestätige mit OK: 
4. Lade dir die alle .py Dateien aus dem GitHub Repository: <https://github.com/patzwalf/led-cube/tree/main>
5. Drücke “CTRL + O“ oder wähle im oberen linken Menü die Funktionen öffnen aus. Jetzt solltest du folgendes Fenster sehen:



Wähle hier „This Computer“ und öffne die .py Dateien. Jetzt kannst du die Dateien in Ruhe ansehen und dich mit dem Code vertraut machen.

1. Speichere den Code auf dem Raspberry Pi. Hierfür drückst du entweder “CTRL + S“ oder wählst oben links speichern. Dann fragt dich Thonny nach dem Speicherort.   
     
   Wähle hier Raspberry Pi Pico aus, um die Dateien auf dem Pico abzuspeichern. Ab jetzt startet der Pico, sobald er mit Strom versorgt ist, die main.py Datei.

# Die Code Base

## main.py

Initialisiert die für den Aufbau **spezifischen LED Objekte und spezifiziert die GPIO PINs** die für die Steuerung genutzt werden.

Das dazugehörige **Dictionary** „dice“ enthält die zu aktivierenden LEDs um eine gewählte Zahl darzustellen.

Außerdem wird ein Knopf zugewiesen, um mit dem Gerät zu interagieren.

Dann folgt vor dem Loop die **Startanimation**, welche dazu dient, die Funktionsfähigkeit aller LEDs sicherzustellen durch einen Durchlauf aller Zahlen bis zur höchsten darstellbaren Zahl.

Der Loop unterscheidet zwischen

1. Dem Drücken des Knopfes, was die Zeit des letzten Knopfdrucks zurücksetzt und dann eine neue pseudozufällige Zahl generiert und darstellt.
2. Einem Loopdurchlauf ohne Knopfdrück. Dann wird geprüft ob der letzte Knopfdruck mindestens 20.000ms (20 Sekunden) zurückliegt und falls die Auswertung TRUE ist wechselt der Pico in einen experimentellen Energiesparmodus.

## lowpower.py – Energiesparen

Da MikroPython aktuell nicht die Funktion des RP2040 Prozessors für einen Energiesparmodus nativ unterstützt wurde im Code ein Workaround von tomjorquera (LGPL-3.0 Lizenz) verwendet. Quellenverweis: <https://github.com/tomjorquera/pico-micropython-lowpower-workaround>

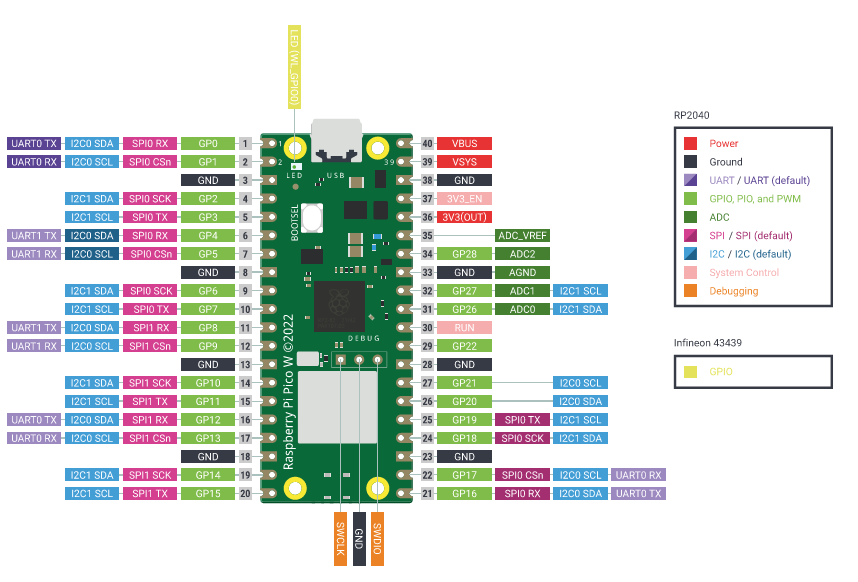
Die Bibliothek erlaubt das Aufrufen von Energiesparfunktionen mit unterschiedlichem Einsparpotential. So kann der Energieverbrauch von 26mA auf 22mA (lightsleep modus) **bis zu 1mA gesenkt werden (deepsleep Modus)**.

Ein Nachteil der Nutzung des deepsleep Modus ist das der PI danach neu bootet. **Vorherige Konfigurationen, die im laufenden Betrieb evtl. erstellt wurden, werden überschrieben / gelöscht.** Im Falle der aktuellen Implementation ist das jedoch zu vernachlässigen.

## dice\_library.py

Die eigens für dieses Projekt erstelle Bibliothek abstrahiert viele Funktionen, die für den Würfel benötigt werden. Um sie korrekt zu verwenden, gilt es ein paar Kleinigkeiten zu beachten.

### Vorbedingungen

1. Die LEDs müssen entsprechend Ihrer GPIO PINs am Mikrocontroller in der richtigen Reihenfolge in einer Liste gespeichert werden. Als Referenz zur Auswahl der PINs das offizielle PINOUT: 

Beispiel:

# Erstellt eine Liste von Pin-Objekten für die LEDs

led\_pins = [16, 17, 18, 19, 20, 21, 22]

leds = [machine.Pin(pin, machine.Pin.OUT) for pin in led\_pins]

1. Die darstellbaren Zahlen als LEDs müssen in der richtigen Reihenfolge in einem Wörterbuch gespeichert werden, dass die LED-Indizes für jede darstellbare Zahl enthalten.

Beispiel:

# Definiert, welche LEDs für jede Zahl leuchten sollen

dice = {

1: [3],

2: [0, 6],

3: [0, 3, 6],

4: [0, 2, 4, 6],

5: [0, 2, 3, 4, 6],

6: [0, 1, 2, 4, 5, 6]

}

### Funktionen

Für alle Parameter gilt:

n = int   
maxDisplayableNumber = int  
leds = Liste mit LED PINs  
dice = Dictionary mit der gewünschten Darstellung welche LED für welche Zahl leuchten soll.

display\_number(n, leds, dice): Zeigt eine bestimmte Zahl auf dem LED-Display an.

dice\_animation(maxDisplayableNumber, leds, dice): Führt eine Würfelanimation aus, indem sie Zahlen von 1 bis maxDisplayableNumber auf dem LED-Display anzeigt.

roll\_dice\_and\_display(maxDisplayableNumber, leds, dice): Generiert eine zufällige Zahl zwischen 1 und maxDisplayableNumber und zeigt sie auf dem LED-Display an.

check\_button\_and\_roll(maxDisplayableNumber, leds, dice): Aktualisiert die Zeit des letzten Tastendrucks für den Schlafmodus und führt die Würfelanimation aus und zeigt eine zufällige Zahl an.

# Funktionsprüfung und Validierung

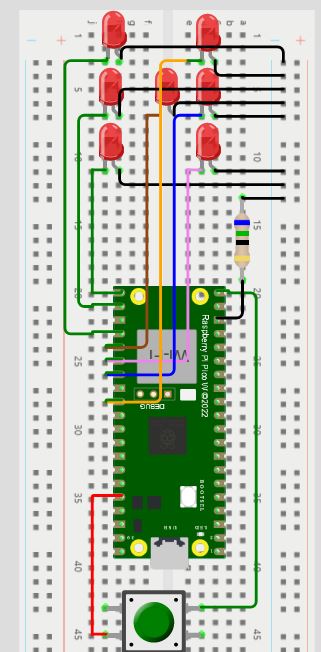
Da es sich um ein Embedded System handelt ist eine Prüfung der Funktionalität in der IDE nicht möglich. Die Kompilierung des Codes erfolgt erst auf dem Raspberry PI was uns vor die Herausforderung stellt das wir nur durch strukturiertes Abfragen eine Validierung der Funktionalität erreichen können.

Hierfür bietet sich folgendes vorgehen an:

1. Wenn mit einem neuen Mikrokontroller gearbeitet wird kann das Standardprogramm „blink“ genutzt werden um die grundsätzliche Funktionsfähigkeit zu überprüfen.
2. Wenn Schritt 1. erfolgreich war kann die main.py geprüft werden. Hierfür bietet es sich an das nach außen sichtbare Verhalten zu beobachten. Dafür muss ein erwartetes Verhalten definiert werden. Im Falle des smarten Würfels ist dies die Initialisierungsanimation. In dieser werden alle LEDs abgelaufen und so kann die korrekte Darstellung der Zahlen sowie die richtige Zuordnung der LEDs geprüft werden.
3. Zum testen der Funktionalität des Schlafmodus kann die Technik des printf-debuggings genutzt werden. Dafür lässt man den Mikrokontroller kurz bevor er den gewünschten Befehl ausführt an die Konsole einen aussagekräftigen Satz ausgeben („Schlafmodus initialisiert“).
4. Der Mikrokontroller sollte jetzt alle LEDs abschalten und in einen Energiesparmodus fallen. Dieser kann geprüft werden mittels Strommessung durch ein Multimeter.
5. Auf Knopfdruck sollte der Würfel wieder in den gewünschten Zustand übergehen und mit dem Würfeln beginnen. Diese soll so aussehen das der Würfel solange würfelt wie der Knopf gedrückt wird. Wenn dieser losgelassen wird, wird die Animation bis zum Schluss abgespielt und eine pseudozufällige Zahl angezeigt.

# Fehlerbehebung

Sollten die **LEDs nicht wie erwartet aufleuchten oder der Knopf nicht auf Knopfdrücke reagieren** bietet es sich an die Kabel und anderen Komponenten zu überprüfen. Diese könnten entweder:  
  
- Locker sein 🡪 Kontrolle der Steckverbindung  
- Beschädigt sein 🡪 Austauschen  
- Nicht an der korrekten Position eingesteckt 🡪 Prüfen mittels Abbildung



# Zukünftige Entwicklung

## Hardware

1. Übergang zu einer PCB um die Kabelverbindungen zu eliminieren um die Zuverlässigkeit zu steigern.
2. Schönere Darstellung der Würfelform durch Erstellung einer Verkleidung.
3. Anpassung der Widerstände um die LED Leuchtkraft zu erhöhen / Potentiometer um diese selbst steuern zu können.

## Software

1. Update auf offizielle Energiesparfunktion sobald diese verfügbar ist.
2. Würfeln auslösen über einen Webserver statt Knopf.
3. Umstellung auf zufällige statt pseudozufälliger Zahlengenerierung. Hierfür könnten die Analogen PINs des Mikrokontrollers ausgelesen werden um diese Werte schwankenden Werte als Seed für die Zahlengenerierung zu nutzen.

# Ressourcen zur Entwicklung

Repository: <https://github.com/patzwalf/led-cube>  
Energiesparmodus: <https://github.com/tomjorquera/pico-micropython-lowpower-workaround>

Dokumentation Raspberry PI Pico W: <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>

Dokumentation Raspberry PI Pico MicroPython SDK: <https://datasheets.raspberrypi.com/pico/raspberry-pi-pico-python-sdk.pdf>  
  
MicroPython für den Pico W: <https://micropython.org/download/RPI_PICO_W/>

# Kontakt

Fabian Patzwall (Entwickler)

Mail: stud106081@fh-wedel.de